Machine Learning Week 8

Unsupervised Learning

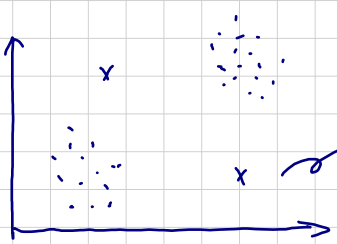
1. Clustering

group / clustering: find <u>some</u> structures

Training set $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

Applications
- market segmentation
- social network analysis
- organize computer clusters
- astronomical data analysis

2. K-means Algorithm.



cluster centroids. (randomly initiated)

step (1) Cluster assignment.
- binary assignment. of datasets. depending on proximity.

(2) Move centroid.
- move to "mean" of location in all labelled

re-colour.

[Input].

① k (number of clusters)

② Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ [drop $x_0 = 1$ convention]

[Algorithm]

Randomly initialize K cluster centroids μ_k. $k = 1 \cdots K$

do {

for $i = 1 \cdots m$.

cluster assignment ($c^i := \min_k \| x^{(i)} - \mu_k \|^2$

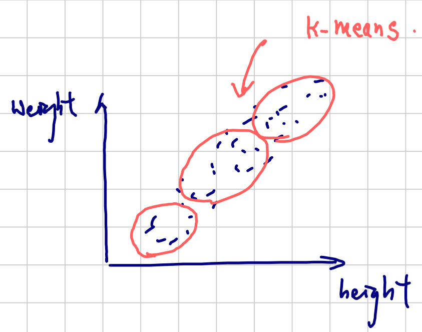
for $k = 1 \cdots K$

move centroid ($\mu_k :=$ mean (pts assigned to cluster k)

$\in \mathbb{R}^n$

3. K-means for non-separated clusters.

T-shirt sizing



Optimization objective

K-means optimization objective.

- c^{i} : index.
- μ_k : cluster centroid k $\in \mathbb{R}^n$
- μ_{c^i} = cluster centroid of cluster to which $x^{(i)}$ has been assigned.

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1 \cdots \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$

$$\min_{\substack{c^{(1)} \dots c^{(m)} \\ \mu_1 \dots \mu_K}} J(\dots)$$

min.

$x^{(i)}$

x μ.

[Algorithm]

Randomly initialize K cluster centroids μ_k , $k = 1 \cdots K$

do {

cluster assignment

for $i = 1 \cdots m$.

$$c^i := \min_k \| x^{(i)} - \mu_k \|^2$$

\# Minimize $J(\cdots)$ wrt. $c^{(1)} \cdots c^{(m)}$. [holding $\mu_1 \cdots \mu_K$]

move centroid

for $k = 1 \cdots K$

$\mu_k :=$ mean (pts assigned to cluster k)

$\in \mathbb{R}^n$

\# Min. w.r.t. $\mu_1 \cdots \mu_K$

Random Initialization

Rules.

(1) $K < m$

(2) Randomly pick K training examples

(3) set $\mu_k =$ examples

- Might have different clustering (local optimum) → try different random initialization.

- Implementation.

$$
\text{for } i = 1 \cdots 1000 \ \{
$$

randomly initialize K-means
run K-means . get $c^{(1)} \cdots c^{(m)}, \mu_1 \cdots \mu_k$

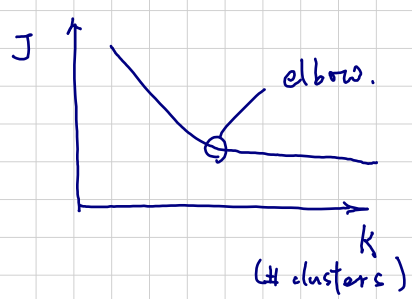
complete cost function (distortion)
↳ $J(c^{(1)}, \ldots, c^{(m)}, \mu_1 \cdots \mu_K)$

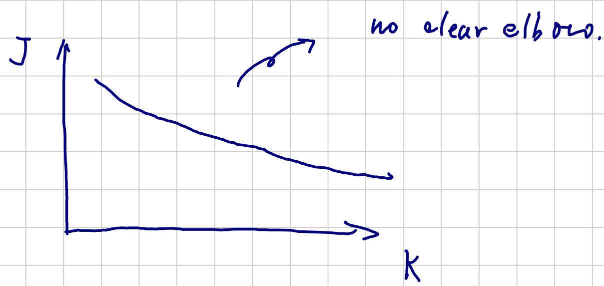
$\}$

Pick one with lowest cost J.

Choose K

1. Elbow method:



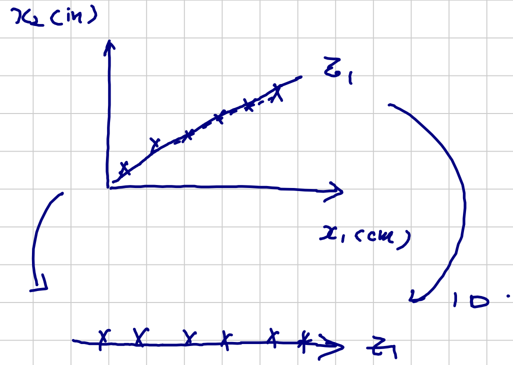


2. Pre-defined (t-shirt sizes : 3 ... 5).

Dimensionality Reduction

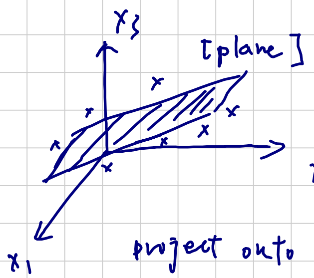
1. Motivation I : Data compression

- Redundent data dimension (cm, inch, ...)

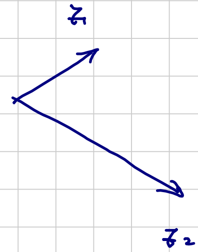


ex. 3D → 2D .

$x^{(i)} \in \mathbb{R}^3$



→ need 2-axis for plane



project onto a plane

$z^{(i)} \in \mathbb{R}^2$. $\begin{pmatrix} z_1^{(i)} \\ z_2^{(i)} \end{pmatrix}$

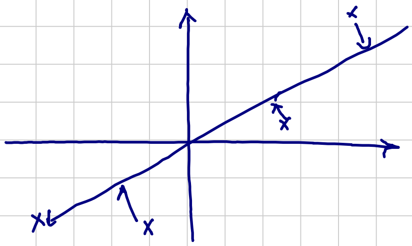
(correlation approximation)

procedure. project data to one less dimension. → re-coordinate.

2 Motivation II : Data visualization.

- Combine certain features (correlated) . $\underset{(500)}{x} \rightarrow \underset{(2)}{z}$

Principal Component Analysis



Try to find a lower dimension surface to min. projection error.

→ perform mean normali. and feature scaling.

· Principal Component Analysis [PCA] problem formulation

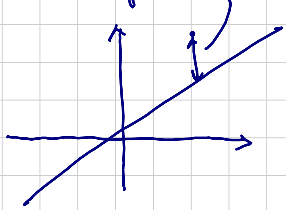
2 → 1
- find dir. vector $u^{(1)} \in \mathbb{R}^n$ onto which data projects

$n \to k$

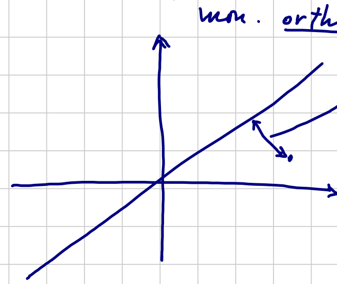
· find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which data projects.

[project data to the linear subset span by $\{u^{(i)}\}_1^k$]

· linear regression
min. squared error



PCA.
min. orthogonal distance



Applys to higher dimension.

Principal component analysis component.

- Data preprocessing
  - mean normalization
  - feature scaling

- Reduction $x^{(i)} \in \mathbb{R}^2 \longrightarrow z^{(i)} \in \mathbb{R}$ [2D to 1D]

- Mathematical derivation (complicated)

<u>Algo.</u>

① Reduce data from $\mathbb{R}^n$ to $\mathbb{R}^k$

Sigma. $\in \mathbb{R}^{n\times n}$

$(n\times 1)\ (1\times n)$

Compute "covariance matrix" $\Sigma = \frac{1}{m} \sum_{i=1}^{n} [x^{(i)}][x^{(i)}]^T = \left(\frac{1}{m}\right) \cdot X^T \cdot X.$

$X = \left(\overset{n}{\equiv} \right)$

Compute "eigenvectors" of matrix Σ

→ singular value decomposition

$[U, S, V] = \text{svd}(\text{Sigma})$

$$U = \begin{bmatrix} | & | & & | \\ \underbrace{u^{(1)} \quad u^{(2)}} & & \cdots & u^{(m)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

first k vectors: $u^{(1)} \cdots u^{(k)}$

② Projection.

$$U_{reduce} = \underbrace{\begin{bmatrix} | & & | \\ u^{(1)} & \cdots & u^{(k)} \\ | & & | \end{bmatrix}}_{\in \mathbb{R}^{n\times k}}$$

$$z^{(i)} = U_{rd}^T \cdot x^{(i)} = \underbrace{\begin{pmatrix} — u^{(1)} — \\ \vdots \\ — u^{(k)} — \end{pmatrix}}_{(k\times n)} \underbrace{x^{(i)}}_{(n\times 1)} = (k\times 1)$$

Reconstruction from Compressed Representation

$z = U_{red}^T x.$ $\qquad z \in \mathbb{R} \xrightarrow[?]{} x \in \mathbb{R}^2$

$x \approx X_{approx} = \underbrace{U_{reduce}}_{(n\times k)} \cdot \underset{(k\times 1)}{z}.$

Choosing k. (# of principal component)

- Average Square projection error $\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - x^{(i)}_{approx} \|^2$
- Total variation in data $\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} \|^2$

- Typically, choose k to be min s.t.

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - x^{(i)}_{approx} \|^2}{\frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} \|^2} \le 0.01 \quad (1\%) \qquad [*]$$

"99% variance retained"

- Algorithm.

1) try PCA with $k = 1$

2) Compute U_{reduce}, $z^{(1)}, z^{(2)}, \dots z^{(m)}$
$x^{(1)}_{approx} \dots x^{(m)}_{approx}$

3) check [*]

4) $k = k+1 \to 2$

- $[U, S, V] = svd(sigma)$

$\underset{n\times n \text{ diagonal}}{S}$, $\quad S = \begin{pmatrix} S_{11} & & 0 \\ & \ddots & \\ 0 & & S_{nn} \end{pmatrix}$

For given k, (*) can be computed as $\underbrace{\left(1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ij}}\right)}_{> 0.99}$

Advice for applying PCA

Supervised Learning Speed Up

$x^{(i)} \in \mathbb{R}^{10000}$

$(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabelled dataset $x^{(1)}, x^{(2)} \ldots x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow$ PCA (defined mapping only on training set)

$z^{(1)} \ldots z^{(m)} \in \mathbb{R}^{1000}$

New training set

$(z^{(1)}, y^{(1)}) \ldots (z^{(m)}, y^{(m)}) \xrightarrow[\text{train.}]{} h_\theta(z) = \dfrac{1}{1+\exp(-\theta^T z)}$

Example

$x \to z \to h(z)$ (prediction)

Application

- Compression $\to$ choose k by % variance retain.
- Visualization $\to$ $k = 2, 3$

What not to do $\to$ prevent overfitting

- Use $z^{(i)}$ instead of $x^{(i)}$. reduce # of features from n to k.

  $\longrightarrow$ fewer features, less likely to overfit. (Why $\to$ PCA throws away information w/o knowing y?

- Should use λ.

PCA usage.

Design of ML sys.?

- get training set

[- run PCA reduce $x^{(i)} \rightarrow z^{(i)}$] do w/o this step first.

- train logistic.

- test set